EP 34717 (1)

Proceedings of the 2001 IEEE/RSJ
International Conference on Intelligent Robots and Systems
Maui, Hawaii, USA, Oct. 29 - Nov. 03, 2001

Control of Robots with Elastic Joints based on Automatic Generation of Inverse Dynamics Models

Michael Thümmel, Martin Otter and Johann Bals
Institut für Robotik und Mechatronik
DLR Oberpfaffenhofen
D-82234 Wessling, Germany
email: Michael.Thuemmel@dlr.de, Martin.Otter@dlr.de, Johann.Bals@dlr.de

Abstract

In this article it is shown how tool vibrations of elastic joint robots can be reduced significantly by feedforward control based on inverse dynamics models. The key ideas are (a) to use available algorithms for differential-algebraic equation systems to derive inverse plant models automatically and reliably, (b) to automatically construct the high-order differentiability of the inverse plant model inputs by prefilters, and (c) to use a controller with two structural degrees of freedom. The technique is demonstrated with a 6 axes elastic joint robot model and with experimental results from our laboratory robot.

1 Introduction

A large number of algorithms are known to control robots under the assumption of rigid joints and rigid links. Controllers designed under this assumption limit the performance because the specialized gears used for robots, such as Harmonic Drives, have a significant elasticity. In particular, the vibrations measured at the tool are considerably higher as predicted with rigid joint models. This behaviour can be easily explained with the following simple experiment:

If only one axis of an elastic joint robot is moving so that gravity has no effect, the mechanical part of the robot can be described by the model in Fig. 1. It consists of the moments of inertia of the motor J_2 and the load J_1 (= the inertia of all outer joints, arms and loads projected on the axis of rotation), a spring/damper combination describing the elasticity k and damping d in the gearbox, velocity dependent friction torques f_1 and f_2 in the bearings of the load and the motor, as well as the motor driving torque τ. All variables on the motor side are scaled by the gear ratio i (e.g. $J_2 := J_{motor} \cdot i^2$; $\Theta := \Theta_{motor}/i$) to eliminate the gear ratio equations for a simpler overall description.

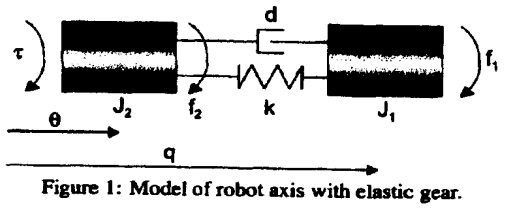

Figure 1: Model of robot axis with elastic gear.

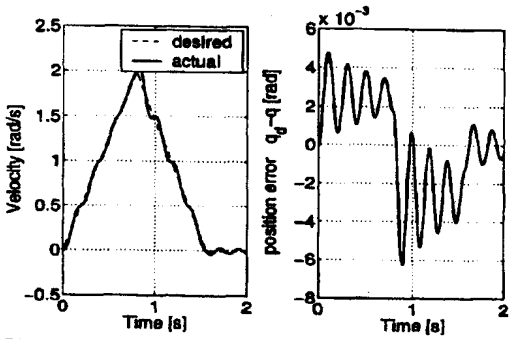

Figure 2: Load dynamics with optimal motor controller.

A controller should keep the deviation between the load angle q and the desired load angle $q_d(t)$ small. Most robots have at least a sensor to measure the motor position Θ. Under the assumption of a rigid joint the desired motor angle $\Theta_d = q_d$ and therefore the motor position error is computed as $e = \Theta_d - \Theta$. Assuming that we have the best possible motor controller which is able to reduce the motor error to zero all the time, the load will still vibrate due to the elasticity.

Typical simulation results are presented in Fig. 2. In the left part the desired speed and the actual speed of the load are shown. In the right part the load position error is displayed. Obviously, even with the best possible motor controller which is based on a rigid joint model, significant oscillations occur, because the load can vibrate even if the controlled motor is at rest.

2 Dynamic Models of Elastic Joint Robots

The equations of motion for a robot having rigid links and flexible joints are well known and a derivation can be found in many articles, e.g., in [1]. The used models mainly differ in the effects of the drive trains that are taken into account. Contrary to many models the elasticity should be modeled as a nonlinear spring as typical datasheets of gearbox manufacturers show. Another effect often neglected is joint damping, which is important when trying to get a realistic model. In this article, the following model of a robot with n axes is used

$$\begin{aligned} 0 &= B(q)\cdot\ddot{q} + S\cdot\ddot{\Theta} + c(q,\dot{q}) + g(q) + \\ &\quad k(q-\Theta) + d(\dot{q}-\dot{\Theta}) + f_L(\dot{q}) \qquad (1) \\ \tau &= S^T\cdot\ddot{q} + J\cdot\ddot{\Theta} - \\ &\quad k(q-\Theta) - d(\dot{q}-\dot{\Theta}) + f_M(\dot{\Theta}) \qquad (2) \end{aligned}$$

where $q(t)$ is the vector of n joint angles, $\Theta(t)$ is the vector of n motor angles, τ is the vector of n driving torques of the motors, J is the $n \times n$ diagonal matrix of the motor inertias, B is the $n \times n$ inertia matrix, S is the $n \times n$ matrix of inertial coupling between motors and links, c is the vector of Coriolis and centrifugal torques, g is the vector of gravitational torques and $f_L(\dot{q})$ and $f_M(\dot{\Theta})$ represent the friction characteristics of link and motor bearings. The same notation as in [2] is used, except that k_i are nonlinear functions of $(q_i - \Theta_i)$ to describe gear elasticity and d_i are nonlinear functions of $(\dot{q}_i - \dot{\Theta}_i)$ to describe gear damping $(i = 1, \ldots, n)$.

Throughout this paper a model of the 6 axis robot Manutec r3 is used as a benchmark. The structure of the model equations is according to (1,2). Simulations are with respect to a desired path where the robot changes the position of all its links from 0 to 1 rad, starting its movement at time = 0.1 s. The start and end positions of this movement are shown in Fig. 3.

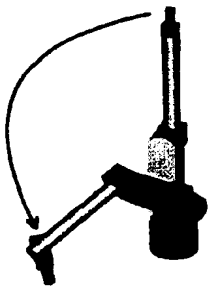

Figure 3: Benchmark motion for simulations with the Manutec r3 robot.

3 Review of Control Algorithms

The development of control strategies for elastic joint robots has received considerable interest in the past years. Overviews can be found in [3, 4]. For example in [2] it is shown that a robot with elasticities in all of it joints is linearizable by dynamic feedback. Unfortunately many of the proposed controllers are difficult to use in practice, e.g., because all robot states have to be measured or large online computations have to be performed. For example, even on the advanced DLR lightweight robot with its large computational power it is not possible to calculate the full feedback linearization algorithm in real time [5]. Due to these reasons, simple controllers like

$$\begin{aligned} \tau &= -K_P\cdot(\Theta - \Theta_d) - K_D\cdot\dot{\Theta} + g(q_d) \qquad (3) \\ \Theta_d &= q_d + k^{-1}(g(q_d)) \qquad (4) \end{aligned}$$

proposed by Tomei in [6] and variations of it are still in widespread use.

The major drawback of this controller is that it exhibits large link vibrations for faster trajectories, as seen in Fig. 4, which is the result of simulations of the Manutec r3 benchmark using the Tomei controller.

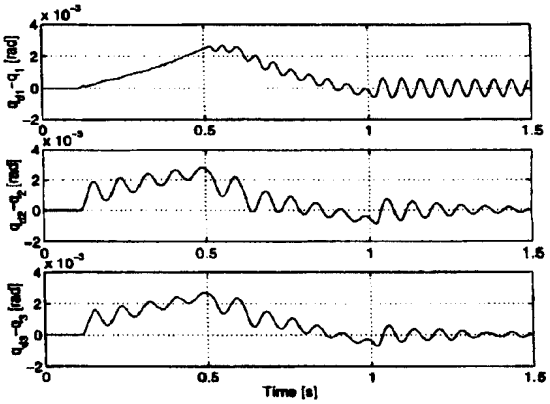

Figure 4: Link errors for benchmark motion using a PD controller and static compensation according to [6].

These vibrations are due to the fact that only torques and static deflections caused by gravitation are compensated in the feedforward part of the controller. This property is sufficient to gain global stability for set point control but results in the oscillating behavior shown in Fig. 4. From this point of view it is straightforward to extend the feedforward part of the controller by taking into account the complete robot dynamics to get less vibrations. This way of calculating the desired trajectories makes sense not only for the PD controller but also for more complex controllers, such as state feedback controllers.

4 Feedforward Control for One Axis

Calculating an input trajectory (here: motor angles Θ_d) for a nonlinear dynamic system corresponding to desired outputs (here: q_d) can be seen as an application of the theory of flat systems, introduced in [7]. Inputs and states of flat systems can be calculated from their flat outputs and a finite number of derivatives of the flat outputs by a system of *algebraic* equations. In [8] this is performed for elastic joint robots assuming constant stiffness and neglecting damping and friction. However, the calculation utilizing only algebraic equations is not always possible as can be shown by the two mass system from Fig. 1 representing one axis and described by the equations:

$$J_1\ddot{q} + f_1(\dot{q}) + k(q-\Theta) + d(\dot{q}-\dot{\Theta}) = 0 \quad (5)$$

$$J_2\ddot{\Theta} + f_2(\dot{\Theta}) - k(q-\Theta) - d(\dot{q}-\dot{\Theta}) = \tau \quad (6)$$

Given $q(t)$ and a sufficient number of derivatives of q, the variables $\Theta(t)$ and $\tau(t)$ shall be calculated from (5,6). When joint damping is neglected, $d(\dot{q}-\dot{\Theta}) = 0$ and Θ can be computed via an *algebraic* equation from (5). However, when damping is taken into account, Θ is determined by the solution of the *differential equation* (5). Afterwards, $\dot{\Theta}$ can be calculated by solving (5) for $\dot{\Theta}$. In addition, (5) needs to be differentiated in order to be able to calculate $\ddot{\Theta}$ which is needed for the evaluation of (6). This leads to a new *algebraic* equation

$$J_1 q^{(3)} + \frac{\partial f_1}{\partial \dot{q}}\ddot{q} + \frac{\partial k}{\partial(q-\Theta)}(\dot{q}-\dot{\Theta}) + \frac{\partial d}{\partial(\dot{q}-\dot{\Theta})}(\ddot{q}-\ddot{\Theta}) = 0 \quad (7)$$

Finally, τ can be calculated from (6) using the previous results. With this example we can already detect some necessary conditions for the application of the proposed method: Due to (7), the characteristic curves of spring, damper and link friction must be continuous and differentiable. Furthermore, the derivatives of the desired link trajectories q_d must be provided up to order three.

5 Inverse System Equations

As shown by example, the derivation of the inverse system equations may require to differentiate certain parts of the model equations and to solve the derived set of nonlinear differential and algebraic equations numerically.

It turns out that the problem of determining the inverse system equations is closely related to solve general DAEs (= differential-algebraic equation systems), as pointed out in [9, 10]. The remaining section will show the relationship formally and sketch the algorithms to transform DAEs into state space form, to automatically determine inverse models.

A system of differential and algebraic equations has in general the following structure:

$$0 = f(\dot{x}, x, y, u) \quad (8)$$

where $x(t)$ are variables which appear differentiated in the model (i.e., $\dot{x}$ is present), $y(t)$ are algebraic and the inputs $u(t)$ are known functions of time t. This system can be transformed to state space form (at least numerically)

$$\begin{bmatrix} \dot{x} \\ y \end{bmatrix} = f_1(x, u) \quad (9)$$

by purely algebraic transformations, if (8) can be solved for $\dot{x}$ and y. Due to the implicit function theorem, this is the case if the Jacobian with respect to these variables is regular:

$$\left| \frac{\partial f}{\partial \dot{x}} \vdots \frac{\partial f}{\partial y} \right| \neq 0 \quad (10)$$

If (10) does not hold, it may still be possible to transform (8) to a state space form, where the number of states is a *subset* of the elements of x. This requires to differentiate certain equations of (8) and to select an appropriate subset of x. For both tasks algorithms are available: The equations to be differentiated can be determined with the algorithm of Pantelides [11] and the selection of state variables can be performed with the "dummy derivative method" of Mattsson and Söderlind [12]. Both algorithms are, e.g., available in the programs Dymola [13] and ABACUS [14].

Deriving an inverse model can now be treated by just exchanging the meaning of variables in a DAE: n_{inv} previously *unknown* variables x_{inv} or y_{inv} are treated as *known* input functions, and n_{inv} previously *known* input functions u_{inv} are treated as *unknown* algebraic variables, i.e., the meaning of input and output variables is exchanged. In any case, the result is still a DAE which can be handled with the same methods as any other DAE, especially with the method of Pantelides and the dummy derivative method.

Since the Pantelides algorithm will differentiate equations, the known input functions may be differentiated too which leads to the well-known effect that the derivatives of the input functions must exist and be provided analytically up to a certain order, which is automatically determined by the algorithm of Pantelides. For this purpose several approaches exists. The two most intuitive are polynomials and using lowpass filters of appropriate order. Below, the latter method is used utilizing Butterworth filters:

$$q_{fd} = \frac{c_0}{a_0 + a_1 s + \ldots + a_{p-1}s^{p-1} + s^p} q_d \quad (11)$$

By using the representation of the filter in controller canonical form, the derivatives of the filter output can be easily computed, since they are the filter states x_F:

$$\dot{x}_F = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & & 1 & & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & \ldots & & 0 & 1 \\ -a_0 & -a_1 & & & -a_{p-1} \end{bmatrix} x_F + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} q_d$$

$$q_{fd} = \begin{bmatrix} c_0 & 0 & 0 & 0 & 0 \end{bmatrix} x_F \quad (12)$$

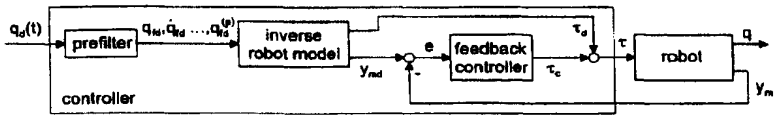

Figure 5: Controller with two structural degrees of freedom.

Hence, the method consists of two steps: (1) using a standard algorithm for link trajectory generation $q_d(t)$ such as "minimum travel time", (2) utilizing filters of appropriate order in the form of (12) to calculate the desired smooth link angles q_{fd} and their needed derivatives.

Using the described approach is only applicable, if the inverse plant model is a *stable* system. For *linear* plant models this means that the plant shall not have *unstable system zeros*, i.e., the plant needs to be a minimum phase system [15]. For general nonlinear plant models it is usually difficult to proof that the inverse plant model is stable and therefore often only via simulations some justification and confidence can be obtained.

6 Controllers with Two Structural Degrees of Freedom for Nonlinear Dynamic Systems

After designing the inverse dynamics model, the question arises how to combine it with a feedback controller. In [15] Kreisselmeier has proposed and analyzed a controller structure with two structural degrees of freedom for linear, single-input/single-output systems. The generalization of this controller structure to *nonlinear* multi-input/multi-output systems is shown in Fig. 5.

As discussed, the desired joint angles q_d of the robot are filtered with a *prefilter*, resulting in the filtered joint angles q_{fd} and their derivatives $\dot{q}_{fd}, \ddot{q}_{fd}, \ldots$ as input to the *inverse robot model*. The inverse robot model computes (a) the desired values y_{md} for all measured quantities y_m used in the *feedback controller* and (b) an additional feedforward torque τ_d. In case of a PD controller Θ and $\dot{\Theta}$ are the measured quantities and therefore $y_{md} = [\Theta_{fd}; \dot{\Theta}_{fd}]$.

The *feedback controller* needs to be designed in such a way that the closed loop system is globally asymptotic stable. Assuming the inverse plant model computes the same value for y_{md} as the measured quantity y_m, the control error e is zero. If the feedback controller starts from a stationary value and the state of the controller is selected in such a way that the outputs vanish, the inputs to the robot are the torques τ_d computed in the inverse plant model. If the inverse plant model is identical to the robot model, the robot will produce $y_m = y_{md}$ (because the inverse plant model was constructed in this way) and the preliminary assumption of zero control error is fulfilled. In other words, the controller has no effect, as long as the plant and the inverse plant model are identical, they both start at the same initial values and both systems are stable.

If this is not the case, a control error occurs and the controller has to stabilize the system and cope with the imprecise inverse plant model. This controller structure is advantageous because the design of the feedback control to stabilize the plant is decoupled from the feedforward control that is responsible for following the given reference input. Note, that the *prefilter* determines essentially the input/output behaviour $q_d \Rightarrow q$ of the overall system, since the inverse plant model and the plant model cancel each other, and therefore it can be interpreted as the "desired transfer function" of the closed loop system.

7 Simulation Results

A model of the Manutec r3 robot has been implemented, including the 3-dimensional mechanics and drive trains with flexibility in every joint, using the free, object-oriented modeling language Modelica [16, 17]. Simulations have been performed with the Dymola program [13]. The highest level of the robot model is shown in Fig. 6.

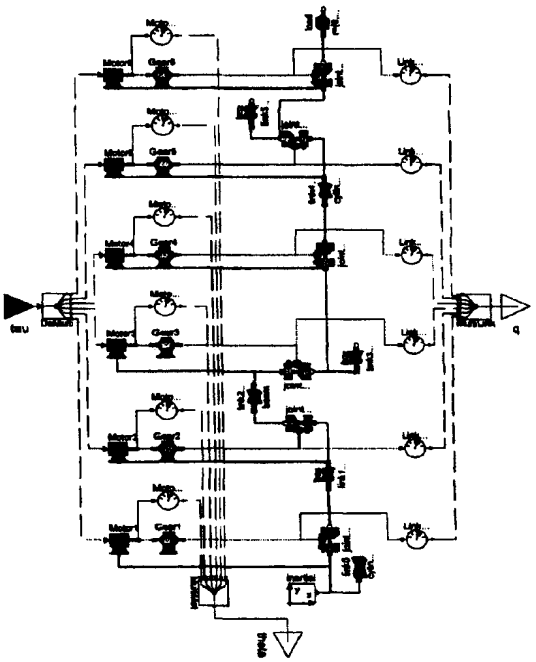

Figure 6: Modelica model of the Manutec r3 Robot.

On the left side, the motor driving torques are provided as an input signal vector which is split into inputs to the 6 motors of the robot. The motors are mounted on the appropriate parts of the robot. The output shafts of the motors are rigidly connected to the gearboxes, which contain elasticity, damping and friction. The output shafts of the gearboxes are rigidly attached to the driving axes of the revolute joints of the mechanical part. Finally, the angles of the joints are extracted, collected and provided as one output signal vector on the right side of the figure. This model of the robot is used for simulations as well as for deriving the inverse robot model. Fig. 7 shows the inverse robot model.

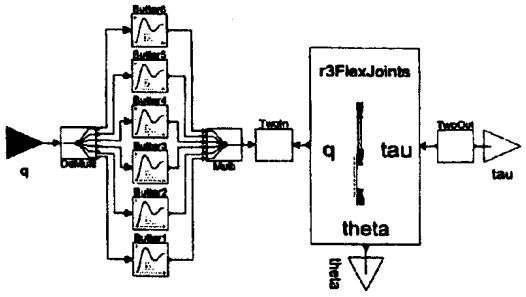

Figure 7: Modelica model of inverse dynamics.

In the right part of this figure, the model of the robot from Fig. 6 is used. In order to derive the inverse robot model, *only the inputs and outputs* of the robot model have to be exchanged, which is achieved by the TwoIn and TwoOut blocks. As a result, the joint angles q are input signals to the robot model and the motor torques are output variables. In addition, the joint angles and their derivatives need to be defined. This is accomplished by Butterworth filters as explained in Section 5, see left part of Fig. 7. Translating this model with Dymola, produces the inverse robot model. Figure 8 shows the feedforward torques of the first three axes computed for the benchmark example.

The inverse robot model is used in combination with a PD controller in simulations to verify the performance improvement. As expected the path deviations are zero except for small numerical errors, if exactly the same model is used for the plant and the inverse plant model. A more interesting question is whether the feedforward control is still useful when the inverse robot model deviates from the robot model. For a simple test, simulations have been performed with the robot load being half and double the expected load. The results are shown in Fig. 9. When comparing Fig. 4 with Fig. 9 it can be seen that the path errors with the unprecise inverse robot model are still one order of magnitude smaller than those caused by the PD controller using only static compensation.

A drawback is the quite large amount of computing time that is needed to calculate the inverse dynamics. On a Pentium III with 700 Mhz it takes about 12 seconds to calculate the feedforward control for a trajectory lasting 1 second with integration algorithm DASSL. The major reason is, that a detailed model with damping is treated and that the motors are mounted on the links and are not approximated to be placed on ground, as it is often assumed. As a consequence, part of the equations of motion need to be differentiated up to order *eight*. Furthermore, nonlinear differential equations need to be solved. When taking into account that the code generated by Dymola is highly optimized, it seems difficult to evaluate the complete dynamics model online. However, it is possible to compute the time consuming part of the feedforward control off-line since it is solely based on the known desired inputs, save the result and use $[y_{md}, \tau_d]$ as inputs to the feedback controller.

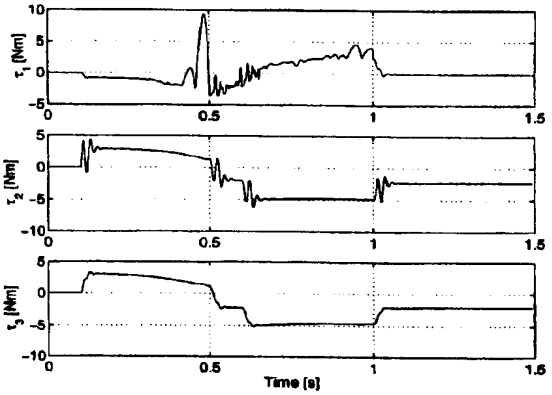

Figure 8: Feedforward torques of the first three axes computed for the benchmark example.

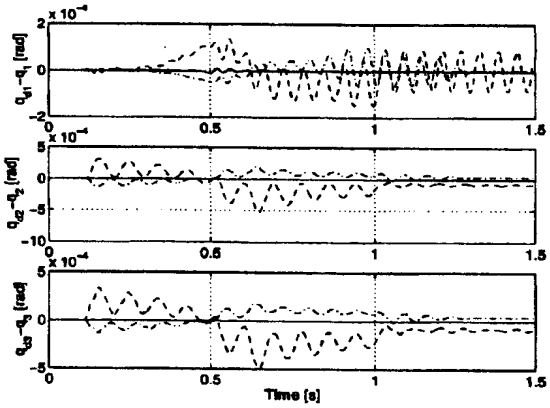

Figure 9: Link errors for Benchmark motion with inverse robot model. Full: optimal model, dashed: double load, dash-dot: half load.

8 Experimental results

In order to verify the results obtained through simulation, tests with our laboratory robot have been carried out. The desired trajectory was chosen to be a square with start point $[-100, 0, 0]$mm traversed in clockwise direction. Figure 10 shows absolute deviations from the desired path displayed on the z axis. As can be seen the deviations are reduced significantly compared to the controller without the inverse robot model.

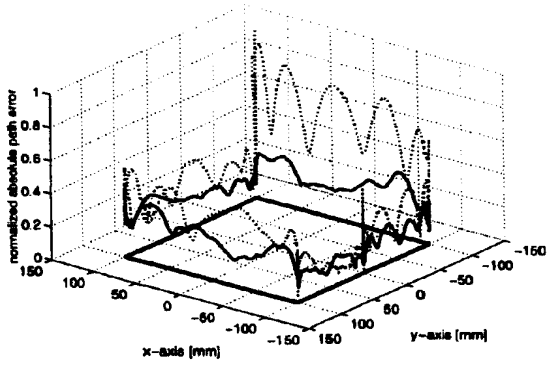

Figure 10: Path deviations with controller using inverse robot model compared to standard controller.

9 Conclusions

This paper pointed out that controllers for elastic joint robots can be improved considerably by extending them with an inverse dynamics feedforward controller. It turns out that even very complex inverse dynamics models can be generated automatically and reliably using algorithms for handling of differential-algebraic equations e.g. available in the program Dymola. Simulations and experimental results show the effectiveness of the proposed method.

References

[1] A. De Luca and P. Tomei. Elastic joints. In C. Canudas de Wit, B. Siciliano, G. Bastin, editor, *Theory of Robot Control*, pages 179–217. Springer-Verlag, Berlin, 1996.

[2] A. De Luca and P. Lucibello. A general algorithm for dynamic feedback linearization of robots with elastic joints. *IEEE Conference on Robotics and Automation*, pages 504–510, 1998.

[3] H. G. Sage, M. F. De Mathelin and E. Ostertag. Robust control of robot manipulators: a survey. *International Journal of Control*, 72(16):1498–1522, 1999.

[4] B. Brogliato, R. Ortega and R. Lozano. Global tracking controllers for flexible-joint manipulators: a comparative study. *Automatica*, 31(7):941–956, 1995.

[5] A. Albu-Schaeffer and G. Hirzinger. State feedback controller for flexible joint robots: A globally stable approach implemented on DLR's light-weight robots. *IEEE Conference on Intelligent Robots and Systems*, 2000.

[6] P. Tomei. A simple PD controller for robots with elastic joints. *IEEE Transactions on automatic control*, 36(10):1208–1213, 1991.

[7] Michel Fliess, Jean Levine, Philippe Martin and Pierre Rouchon. Flatness and defect of non-linear systems: introductory theory and examples. *International Journal of Control*, 61(6):1327–1361, 1995.

[8] A. de Luca. Feedforward / feedback laws for the control of flexible robots. *IEEE Conference on Robotics and Automation*, pages 233–240, 2000.

[9] F. Mugica and F.E. Cellier. Automated synthesis of a fuzzy controller for cargo ship steering by means of qualitative simulation. In *Proceedings of the European Simulation MultiConference (ESM'94)*, pages 523–528, Barcelona, Spain, 1994.

[10] M. Otter and F.E. Cellier. Software for modeling and simulating control systems. In W.S. Levine, editor, *The Control Handbook*, pages 415–428. CRC Press, 1996.

[11] C.C Pantelides. The consistent initialization of differential-algebraic systems. *SIAM Journal of Scientific and Statistical Computing*, pages 213–231, 1988.

[12] S.E. Mattsson and G. Söderlind. Index reduction in differential-algebraic equations using dummy derivatives. *SIAM Journal of Scientific and Statistical Computing*, pages 677–692, 1993.

[13] Dymola. Homepage: http://www.dynasim.se/.

[14] ABACUS. Homepage: http://yoric.mit.edu/abacuss.

[15] G. Kreisselmeier. Struktur mit zwei Freiheitsgraden. *Automatisierungstechnik at 6*, pages 266–269, 1999.

[16] S.E: Mattsson H. Elmqvist and M. Otter. Modelica - a language for physical system modeling, visualization and interaction. In *1999 IEEE Symposium on Computer-Aided Control System Design*, 1999.

[17] Modelica Association. *Modelica – A Unified Object-Oriented Language for Physical Systems Modeling. Tutorial, Version 1.4*. Modelica Association, http://www.Modelica.org/documents, 2000.